Linear usages in signal based communication

Roberto Amadio   Mehdi Dogguy

Université Paris Diderot
Laboratoire Preuves, Programmes et Systèmes

December 13, 2007

Plan

1 - $S\pi$-calculus

- is a synchronous π-calculus,
- no non-deterministic choice operator +,
- a dereferenciation operator $!s$,
- inputs depend on the instant,
- outputs are persistent within the instant.

Examples

$$pause.K = \nu s(\mathtt{s}(\mathtt{x}).0, K)$$

Examples

$$pause.K = \nu s(\mathtt{s}(\mathtt{x}).0, K)$$

$$A = \nu s, s', s''(\overline{\mathtt{s}}<1> \mid \overline{\mathtt{s}}<2> \mid \mathtt{s}'(\mathtt{x}).\overline{\mathtt{s}''}<1>, 0 \mid pause.\overline{\mathtt{s}''}<!\mathtt{s}>)$$

Examples

$$pause.K = \nu s(\mathtt{s}(\mathtt{x}).0, K)$$

$$A = \nu s, s', s''(\overline{\mathtt{s}}<1> \mid \overline{\mathtt{s}}<2> \mid \mathtt{s}'(\mathtt{x}).\overline{\mathtt{s}''}<1>, 0 \mid pause.\overline{\mathtt{s}''}<!\mathtt{s}>)$$

$$\downarrow N$$

$$A = \nu s, s', s''(0 \mid 0 \mid 0 \mid \overline{\mathtt{s}''}<[1;2]>)$$

Examples

$$pause.K = \nu s(\mathtt{s}(\mathtt{x}).0, K)$$

$$A = \nu s, s', s''(\overline{\mathtt{s}}<1> \mid \overline{\mathtt{s}}<2> \mid \mathtt{s}'(\mathtt{x}).\overline{\mathtt{s}''}<1>, 0 \mid pause.\overline{\mathtt{s}''}<!\mathtt{s}>)$$

$$\downarrow N$$

$$A = \nu s, s', s''(0 \mid 0 \mid 0 \mid \overline{\mathtt{s}''}<[2; 1]>)$$

Within the instant

$$(out)\quad \frac{e \Downarrow v}{\overline{\mathtt{s}}\mathtt{<e>} \xrightarrow{\overline{\mathtt{s}}\mathtt{<v>}} \overline{\mathtt{s}}\mathtt{<e>}} \qquad\qquad (in_{aux})\quad \frac{}{\mathtt{s}(\mathtt{x}).\mathtt{P},\mathtt{K} \xrightarrow{\mathtt{s?v}} [v/x]P}$$

$$(in)\quad \frac{}{P \xrightarrow{\mathtt{sv}} (P \mid \overline{\mathtt{s}}\mathtt{<v>})}$$

$$(comp)\quad \frac{P_1 \xrightarrow{\mu} P_1' \quad bn(\mu) \cap fn(P_2) = \emptyset}{P_1 \mid P_2 \xrightarrow{\mu} P_1' \mid P_2} \qquad (synch)\quad \frac{P_1 \xrightarrow{\nu\mathbf{t}\overline{\mathtt{s}}\mathtt{<v>}} P_1' \quad P_2 \xrightarrow{\mathtt{s?v}} P_2' \quad \{\mathbf{t}\} \cap fn(P_2) = \emptyset}{P_1 \mid P_2 \xrightarrow{\tau} \nu\mathbf{t}(P_1' \mid P_2')}$$

$$(=_1^{ind})\quad \frac{match(v,p) = \theta}{[\mathtt{v} \trianglerighteq \mathtt{p}]\mathtt{P}_1,\mathtt{P}_2 \xrightarrow{\tau} \theta P_1} \qquad\qquad (=_2^{ind})\quad \frac{match(v,p) = \uparrow}{[\mathtt{v} \trianglerighteq \mathtt{p}]\mathtt{P}_1,\mathtt{P}_2 \xrightarrow{\tau} P_2}$$

At the end of the instant

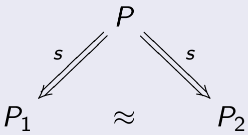
$$(0) \quad \frac{}{0 \xrightarrow{\emptyset, \mathtt{V}} 0} \qquad\qquad (reset) \quad \frac{e \Downarrow v \quad v \in V(s)}{\overline{\mathtt{s}}\langle \mathtt{e} \rangle \xrightarrow{[\{\mathtt{v}\}/\mathtt{s}], \mathtt{V}} 0}$$

$$(cont) \quad \frac{s \notin dom(V)}{s(x).P, K \xrightarrow{\emptyset, \mathtt{V}} V(K)} \qquad (par) \quad \frac{P_i \xrightarrow{\mathtt{E_i}, \mathtt{V}} P'_i \quad i = 1, 2}{(P_1 \mid P_2) \xrightarrow{\mathtt{E_1} \vee \mathtt{E_2}, \mathtt{V}} (P'_1 \mid P'_2)}$$

$$(next) \quad \frac{P \leq \nu s \ P' \quad P' \xrightarrow{\mathtt{E}, \mathtt{V}} P'' \quad V \Vdash E}{P \xrightarrow{\mathtt{N}} \nu s \ P''}$$

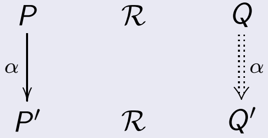
Definition

P is deterministic if for every sequence of actions s :



Definition (Bisimulation Standard)

A symmetric relation $\mathcal{R}$ on programs is a labelled bisimulation if $bn(\alpha) \cap fn(Q) = \emptyset$ and :



We denote with $\approx$ the largest labelled bisimulation.

2 - Typing system 1

- At least two different values are emitted on the same signal, at the same instant :

$$\overline{\mathtt{s}}{<}\mathtt{v}{>} \mid \overline{\mathtt{s}}{<}\mathtt{v}'{>} \mid \mathtt{s}(\mathtt{x}).\mathtt{P}, \mathtt{K}$$

- At the end of the instant, at least two different values are available :

$$\overline{\mathtt{s}}{<}\mathtt{v}{>} \mid \overline{\mathtt{s}}{<}\mathtt{v}'{>} \mid \mathit{pause}.A(!s)$$

$$s : Sig_{\begin{pmatrix} n \\ m \\ p \end{pmatrix}}{}^{\omega}(\sigma)$$

$$\text{où : } \begin{cases} n & \text{max of outputs} \\ m & \text{max of inputs} \\ p & \text{max of inputs at the end of the instant} \end{cases}$$

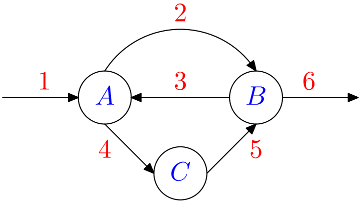
$$n, m, p \in L = \{0 < 1 < \infty\}$$

Lifting operation : $\uparrow (u.v^{\omega}) = v^{\omega}$

Synchronous Data Flow

$$\nu \mathtt{s}_2, \mathtt{s}_3, \mathtt{s}_4, \mathtt{s}_5\ (A(s_1, s_2, s_3, s_4) \mid B(s_2, s_3, s_5, s_6) \mid C(s_4, s_5))$$

$$\begin{array}{lcl}
A(s_1, s_2, s_3, s_4) & = & \mathtt{s}_1(\mathtt{x}).(\overline{\mathtt{s}_2}\mathtt{<f(x)>} \mid \mathtt{s}_3(\mathtt{y}).(\overline{\mathtt{s}_4}\mathtt{<g(y)>} \mid \textit{pause}.A(s_1, s_2, s_3, s_4)), 0), 0 \\
B(s_2, s_3, s_5, s_6) & = & \mathtt{s}_2(\mathtt{x}).(\overline{\mathtt{s}_3}\mathtt{<i(x)>} \mid \mathtt{s}_5(\mathtt{y}).(\overline{\mathtt{s}_6}\mathtt{<l(y)>} \mid \textit{pause}.B(s_2, s_3, s_5, s_6)), 0), 0 \\
C(s_4, s_5) & = & \mathtt{s}_4(\mathtt{x}).(\overline{\mathtt{s}_5}\mathtt{<h(x)>} \mid \textit{pause}.C(s_4, s_5)), 0
\end{array}$$



Cell

$$
\begin{array}{lcl}
\texttt{Cell}(\texttt{q}, \texttt{s}, \texttt{l}) & = & \texttt{Send}(\texttt{q}, \texttt{s}, \texttt{l}, \texttt{l}) \\
\texttt{Send}(\texttt{q}, \texttt{s}, \texttt{l}, \texttt{l}') & = & [\texttt{l}' \trianglerighteq \texttt{cons}(\texttt{s}', \texttt{l}'')] \\
 & & \quad (\overline{\texttt{s}'}\texttt{<q>} \mid \texttt{Send}(\texttt{q}, \texttt{s}, \texttt{l}, \texttt{l}'')), \\
 & & \quad \texttt{pause.Cell}(\texttt{next}(\texttt{q}, !\texttt{s}), \texttt{s}, \texttt{l})
\end{array}
$$

$$[e]\ \begin{pmatrix}\infty\\0\\\infty\end{pmatrix} \qquad [o_1]\ \begin{pmatrix}1\\\infty\\\infty\end{pmatrix}$$

$$[o_0]\ \begin{pmatrix}0\\\infty\\\infty\end{pmatrix}$$

$$\sigma \quad ::= \quad 1 \mid List(\sigma) \mid Set(\sigma) \mid Sig_u(\sigma)$$
$$u \in \{e^\omega, o_0^\omega\}$$

$$\rho \quad ::= \quad \sigma \mid Sig_u(\sigma)$$
$$u \in \{e^\omega, o_1^\omega, o_0^\omega, o_1.o_0^\omega, o_0.o_1^\omega\}$$

Addition

$$\begin{array}{ccccc} 0 & \oplus & x & = & x \\ 1 & \oplus & 1 & = & \infty \\ \infty & \oplus & x & = & \infty \end{array}$$

$$\begin{array}{ccccc} \sigma & \oplus & \sigma & = & \sigma \\ Sig_u(\sigma) & \oplus & Sig_{u'}(\sigma) & = & Sig_{u \oplus u'}(\sigma) \\ \rho & \oplus & \rho' & = & \uparrow \end{array}$$

σ is neutral $\Leftrightarrow \forall \rho, \sigma \oplus \rho = \rho$

... for expressions

$$(var)\quad \frac{\Gamma \text{ neutral}}{\Gamma, x : \rho \vdash x : \rho}$$

$$(k)\quad \frac{\Gamma \vdash r_i : \sigma_i \quad i = 1, \ldots, n \quad \Gamma' \text{ neutral} \qquad k : (\sigma_1, \ldots, \sigma_n) \rightarrow \sigma \quad k = f \text{ or } k = c}{\Gamma' \oplus \Gamma \vdash k(r_1, \ldots, r_n) : \sigma}$$

$$(!_{Set})\quad \frac{\Gamma \text{ neutral} \quad u = (\infty, 0, \infty)^{\omega}}{\Gamma, s : Sig_u(\sigma) \vdash !s : Set_(\sigma)}$$

$$(!_{List})\quad \frac{\Gamma \text{ neutral} \quad u = (0, \infty, \infty)^{\omega}}{\Gamma, s : Sig_u(\sigma) \vdash !s : List(\sigma)}$$

... for programs

$$(0)\quad \frac{\Gamma \text{ neutral}}{\Gamma \vdash 0}$$

$$(out)\quad \frac{u \in \{e^\omega, o_1.o_0^\omega\} \quad \Gamma \vdash e : \sigma}{\Gamma, s : Sig_u(\sigma) \vdash \overline{\mathtt{s}}\mathtt{<e>}}$$

$$(\nu)\quad \frac{\begin{array}{c} u \in \{e^\omega, o_1^\omega\} \\ \Gamma, s : Sig_u(\sigma) \vdash P \end{array}}{\Gamma \vdash \nu s : Sig_u(\sigma)\ P}$$

$$(in)\quad \frac{\begin{array}{c} u \in \{o_0^\omega, o_0.o_1^\omega\} \quad \Gamma \vdash s : Sig_u(\sigma) \\ \Gamma, x : \sigma \vdash P \quad \uparrow(\Gamma) \vdash K \end{array}}{\Gamma \vdash \mathtt{s}(\mathtt{x}).P, K}$$

$$(par)\quad \frac{\Gamma_i \vdash P_i \quad i = 1,2}{\Gamma_1 \oplus \Gamma_2 \vdash P_1 \mid P_2}$$

$$(rec)\quad \frac{\begin{array}{c} A : (\rho_1, \ldots, \rho_n) \\ \Gamma_i \vdash r_i : \rho_i \quad i = 1, \ldots, n \end{array}}{\Gamma_1 \oplus \cdots \oplus \Gamma_n \vdash A(r_1, \ldots, r_n)}$$

$$(m_s)\quad \frac{\Gamma \vdash P_i \quad i = 1,2}{\Gamma \vdash [s_1 = s_2]P_1, P_2}$$

$$(m_c)\quad \frac{\begin{array}{c} c : (\sigma_1, \ldots, \sigma_n) \rightarrow \sigma \quad \Gamma' \vdash u : \sigma \quad \Gamma \oplus \Gamma' = \Gamma \\ \Gamma, x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash P_1 \quad \Gamma \vdash P_2 \end{array}}{\Gamma \vdash [\mathtt{u} \trianglerighteq \mathtt{c}(\mathtt{x}_1, \ldots, \mathtt{x}_n)]\mathtt{P}_1, \mathtt{P}_2}$$

Proposition (Subject Reduction)

Suppose $\Gamma \vdash P$. Then :

(1) If $P \xrightarrow{\mathtt{sv}} P'$, $\Gamma' \vdash \overline{\mathtt{s}}\texttt{<v>}$ and $\Gamma \oplus \Gamma' \downarrow$, then $\Gamma \oplus \Gamma' \vdash P'$.

(2) If $P \xrightarrow{\nu \mathtt{t}:\rho \overline{\mathtt{s}}\texttt{<v>}} P'$ then $\Gamma, t : \rho \vdash P'$.

(3) If $P \xrightarrow{\tau} P'$ then $\Gamma \vdash P'$.

(4) If $P \xrightarrow{\mathtt{N}} P'$ then $\uparrow(\Gamma) \vdash P'$.

Theorem (Typing implies determinacy)

If the program P is typable in a context Γ, then P is determinate.

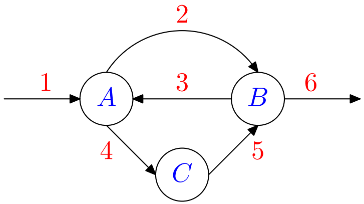
Synchronous Data Flow

$$\nu \mathtt{s}_2, \mathtt{s}_3, \mathtt{s}_4, \mathtt{s}_5\ (\mathtt{A}(\mathtt{s}_1, \mathtt{s}_2, \mathtt{s}_3, \mathtt{s}_4) \mid \mathtt{B}(\mathtt{s}_2, \mathtt{s}_3, \mathtt{s}_5, \mathtt{s}_6) \mid \mathtt{C}(\mathtt{s}_4, \mathtt{s}_5))$$

$$I = Sig_{o_0^\omega}(Data) \qquad O = Sig_{o_1^\omega}(Data)$$

$$A : (I, O, I, O) \qquad B : (I, O, I, O) \qquad C : (I, O)$$

$$s_1 : I \qquad s_6 : O$$



Cell

$$
\begin{array}{lcl}
\texttt{Cell}(q, s, l) & = & \texttt{Send}(q, s, l, l) \\
\texttt{Send}(q, s, l, l') & = & [l' \trianglerighteq \texttt{cons}(s', l'')] \\
& & \quad (\overline{s'}\langle q \rangle \mid \texttt{Send}(q, s, l, l'')), \\
& & \quad \texttt{pause.Cell}(\texttt{next}(q, !s), s, l)
\end{array}
$$

$$S_1 = Sig_{e^\omega}(\sigma) \qquad L_1 = List(S_1)$$

$$Cell : (State, S_1, L_1) \qquad Send : (State, S_1, L_1, L_1)$$

$$next : (State, Set(State)) \rightarrow State)$$

Client-Server

$$
\begin{array}{lcl}
\texttt{Server}(s) & = & \texttt{pause.Handle}(s, !s) \\
\texttt{Handle}(s, l) & = & [l \trianglerighteq cons(req(s', x), l')] \\
 & & \quad (\overline{\texttt{s}'}\texttt{<f(x)>} \mid \texttt{Handle}(s, l')), \\
 & & \quad \texttt{Server}(s) \\
 & & \\
\texttt{Client}(x, s, t) & = & \nu s'(\\
 & & \qquad \overline{\texttt{s}}\texttt{<req(s',x)>} \\
 & & \quad \mid pause.\texttt{s}'\texttt{(x)}.\overline{\texttt{t}}\texttt{<x>}, 0 \\
 & &)
\end{array}
$$

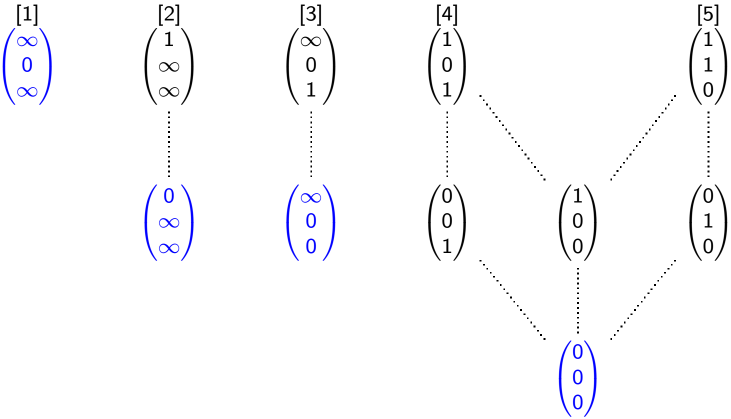
Client-Server

$$
\begin{array}{lcl}
\mathtt{Server}(s) & = & \mathtt{pause.Handle}(s, !s) \\
\mathtt{Handle}(s, l) & = & [l \trianglerighteq cons(req(s', x), l')] \\
 & & \quad (\overline{\mathtt{s}'}\mathtt{<f(x)>} \mid \mathtt{Handle}(s, l')), \\
 & & \quad \mathtt{Server}(s) \\
 & & \\
\mathtt{Client}(x, s, t) & = & \nu s'(\\
 & & \qquad \overline{\mathtt{s}}\mathtt{<req(s', x)>} \\
 & & \quad \mid pause.\mathtt{s}'\mathtt{(x)}.\overline{\mathtt{t}}\mathtt{<x>}, 0 \\
 & &)
\end{array}
$$

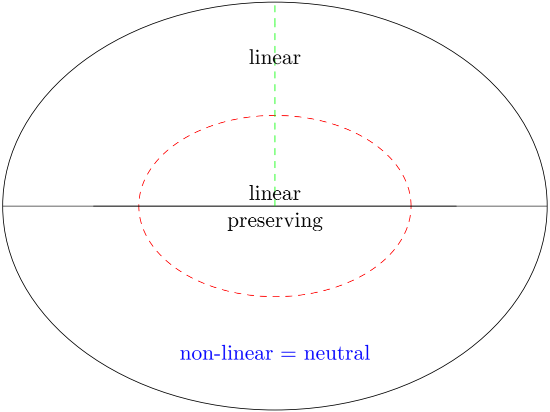
more usages !

3 - Extended system

Signals' classes



Classification



$$
\begin{array}{lll}
\kappa & ::= & 1 \mid \mathit{List}_\infty(\kappa) \mid \mathit{Set}_\infty(\kappa) \mid \mathit{Sig}_u(\kappa) \\
& & \text{u is neutral} \\
\\
\lambda & ::= & \mathit{List}_1(\sigma) \mid \mathit{Set}_1(\sigma) \mid \mathit{Sig}_u(\kappa) \mid \mathit{Sig}_v(\lambda) \\
& & \text{u is uniform and linear} \\
\\
\sigma & ::= & \kappa \mid \lambda \\
& & \text{v is uniform and linear preserving} \\
\\
\rho & ::= & \sigma \mid \mathit{Sig}_u(\kappa) \mid \mathit{Sig}_v(\lambda) \\
& & \text{u is neutral and v is linear}
\end{array}
$$

Addition

$$\begin{array}{ccccc} 0 & \oplus & x & = & x \\ 1 & \oplus & 1 & = & \uparrow \\ \infty & \oplus & x & = & \infty \end{array}$$

$$u \oplus u' \text{ is defined} \quad \Leftrightarrow \quad \left[u, u' \in Class(n) \wedge u \oplus u' \in Class(n)\right]$$

$$\begin{array}{lclcl} \kappa & \oplus & \kappa & = & \kappa \\ Op_\infty(\kappa_1, \ldots, \kappa_n) & \oplus & Op_\infty(\kappa_1, \ldots, \kappa_n) & = & Op_\infty(\kappa_1, \ldots, \kappa_n) \\ Sig_u(\sigma) & \oplus & Sig_v(\sigma) & = & Sig_{u \oplus v}(\sigma) \\ \rho & \oplus & \rho' & = & \uparrow \end{array}$$

... for expressions

$$(var)\quad \frac{\Gamma \text{ neutral}}{\Gamma, x : \rho \vdash x : \rho}$$

$$(k)\quad \frac{\Gamma_i \vdash r_i : \sigma_i \quad i = 1, \ldots, n \quad \Gamma_0 \text{ neutral} \qquad k : (\sigma_1, \ldots, \sigma_n) \to \sigma \quad k = f \text{ or } k = c}{\Gamma_0 \oplus \Gamma_1 \oplus \cdots \oplus \Gamma_n \vdash k(r_1, \ldots, r_n) : \sigma}$$

$$(!_{Set_i})\quad \frac{\Gamma \text{ neutral} \qquad i = \pi_3(u[0]) \qquad u \in U_!(2,3)}{\Gamma, s : Sig_u(\sigma) \vdash !s : Set_i(\sigma)}$$

$$(!_{List_i})\quad \frac{\Gamma \text{ neutral} \qquad i = \pi_3(u[0]) \qquad u \in U_!(2,4)}{\Gamma, s : Sig_u(\sigma) \vdash !s : List_i(\sigma)}$$

... for programs

$$(0)\quad \frac{\Gamma \text{ neutral}}{\Gamma \vdash 0} \qquad (out)\quad \frac{u \in U_{out} \quad \Gamma_1 \vdash s : Sig_u(\sigma) \quad \Gamma_2 \vdash e : \sigma}{\Gamma_1 \oplus \Gamma_2 \vdash \overline{\mathtt{s}}\mathtt{<e>}}$$

$$(\nu)\quad \frac{\Gamma, s : Sig_u(\sigma) \vdash P}{\Gamma \vdash \nu s\ P} \qquad (in)\quad \frac{u \in U_{in} \quad \Gamma_1 \vdash s : Sig_u(\sigma) \quad \Gamma_2, x : \sigma \vdash P \quad \uparrow(\Gamma_1 \oplus \Gamma_2) \vdash K}{\Gamma_1 \oplus \Gamma_2 \vdash \mathtt{s}(\mathtt{x}).P, K}$$

$$(par)\quad \frac{\Gamma_i \vdash P_i \quad i = 1,2}{\Gamma_1 \oplus \Gamma_2 \vdash P_1 \mid P_2} \qquad (rec)\quad \frac{A : (\rho_1, \ldots, \rho_n) \quad \Gamma_i \vdash r_i : \rho_i \quad i = 1, \ldots, n}{\Gamma_1 \oplus \cdots \oplus \Gamma_n \vdash A(r_1, \ldots, r_n)}$$

$$(m_s)\quad \frac{\Gamma \vdash P_i \quad i = 1,2}{\Gamma \vdash [s_1 = s_2]P_1, P_2} \qquad (m_c)\quad \frac{c : (\sigma_1, \ldots, \sigma_n) \to \sigma \quad \Gamma' \vdash u : \sigma \quad \Gamma, x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash P_1 \quad \Gamma \oplus \Gamma' \vdash P_2}{\Gamma \oplus \Gamma' \vdash [\mathtt{u} \trianglerighteq \mathtt{c}(\mathtt{x}_1, \ldots, \mathtt{x}_n)]\mathtt{P}_1, \mathtt{P}_2}$$

Client-Server

$$
\begin{array}{lcl}
\texttt{Server}(s) & = & \texttt{pause.Handle}(s, !s) \\
\texttt{Handle}(s, l) & = & [l \trianglerighteq cons(req(s', x), l')] \\
 & & \quad (\overline{\texttt{s}'}\texttt{<f(x)>} \mid \texttt{Handle}(s, l')), \\
 & & \quad \texttt{Server}(s)
\end{array}
$$

$$
\begin{array}{lcl}
\texttt{Client}(x, s, t) & = & \nu s'(\\
 & & \qquad \overline{\texttt{s}}\texttt{<req(s',x)>} \\
 & & \quad \mid pause.\texttt{s}'(\texttt{x}).\overline{\texttt{t}}\texttt{<x>}, 0 \\
 & &)
\end{array}
$$

$$
s : Sig_{\begin{pmatrix}\infty\\0\\1\end{pmatrix}}^{\omega}(Req) \qquad l : Set_1(Req) \qquad s' : Sig_{\begin{pmatrix}1\\1\\0\end{pmatrix}}^{\omega}(Req)
$$

Client-Server

$$
\begin{array}{lcl}
\texttt{Server}(s) & = & \texttt{pause.Handle}(s, !s) \\
\texttt{Handle}(s, l) & = & [l \trianglerighteq cons(req(s', x), l')] \\
 & & \quad (\overline{\texttt{s}'}\texttt{<f(x)>} \mid \texttt{Handle}(s, l')), \\
 & & \quad \texttt{Server}(s)
\end{array}
$$

$$
\begin{array}{lcl}
\texttt{Client}(x, s, t) & = & \nu s'(\\
 & & \quad\quad \overline{\texttt{s}}\texttt{<req(s}',\texttt{x)>} \\
 & & \quad \mid pause.\texttt{s}'\texttt{(x)}.\overline{\texttt{t}}\texttt{<x>}, 0 \\
 & &)
\end{array}
$$

It's not typable !

Client-Server

$$
\begin{array}{lcl}
\texttt{Server}(s) & = & \texttt{pause.Handle}(s, !s) \\
\texttt{Handle}(s, l) & = & [l \trianglerighteq cons(req(s', x), l')] \\
 & & \quad (\overline{\texttt{s}'}\texttt{<f(x)>} \mid \texttt{Handle}(s, l')), \\
 & & \quad \texttt{Server}(s)
\end{array}
$$

$$
\begin{array}{lcl}
\texttt{Client}(x, s, t) & = & \nu s'(\\
 & & \quad \overline{\texttt{s}}\texttt{<req(s',x)>} \\
 & & \quad \mid pause.\texttt{s}'\texttt{(x)}.\overline{\texttt{t}}\texttt{<x>}, 0 \\
 & &)
\end{array}
$$

Solution ? separate $!s$ from the rest.

4 - Future work

1. `Client-Server` example,
2. subject reduction lemma.

Thank you ...